**Thread**

A thread (or lightweight process) is the basic unit of CPU utilization; it consists of:
- Program counter, register set, and stack space.
- A thread shares with other threads belonging to the same process:
  - Code section, data section, and OS resources (open files, signals).
- Collectively called a task.

If a process has multiple threads of control, it can perform more than one task at a time - multithreading.

**Multithreading Examples**

- A web browser might have one thread display images or text while another thread retrieves data from the network.
- A web server accepts client requests for web pages, images, sounds, etc.
- A word processor may have a thread for displaying graphics, another thread for responding to keystrokes from the user, and a third thread for performing spelling and grammar checking in the background.

**Single and Multithreaded Processes**

- **Benefits**
  - Responsiveness - multithreading an interactive application may allow a program to continue running even if part of it is blocked, for instance: web browser could still allow user interaction in one thread while image was being loaded in another thread.
  - Resource Sharing - allows an application to have several different threads of activity within the same address space.
  - Economy - more economical to create than process creation, because threads share resources.
  - Utilization of multiprocessor Architectures - concurrency.

- **User Threads**

  Thread management done by user-level threads library
  Supported above the kernel, via a set of library calls at the user level.
  Threads do not need to call OS and cause interrupts to kernel - fast.
  Example thread libraries:
  - POSIX Pthreads
  - Win32 threads
  - Java threads
Kernel Threads

Supported by the Kernel

Examples
- Windows XP/2000
- Solaris
- Linux
- Tru64 UNIX
- Mac OS X
- Mach, OS/2

Multithreading Models

Many-to-One

One-to-One

Many-to-Many

Many-to-One

Many user-level threads mapped to single kernel thread

Examples:
- Solaris Green Threads
- GNU Portable Threads

One-to-One

Each user-level thread maps to kernel thread

Examples
- Windows NT/XP/2000
- Linux
- Solaris 9 and later
Many-to-Many Model

- Allows many user level threads to be mapped to many kernel threads
- Allows the operating system to create a sufficient number of kernel threads